リリース 1: データ入力機能の実装

市東 亘

平成 29 年 10 月 25 日

1　概観

目 次

2　必要な機能の洗い出し

データ入力機能.

- カードは階層カテゴリで分類したい.
- カードのデザイン: 表も裏も自由記述.

もう少し細かく機能を洗い出してみよう.
データ入力機能

- カード入力項目: 表，裏，カテゴリ，作成・更新日付（自動）.
- カテゴリ・データ入力項目: カテゴリ名，親カテゴリ.

データ表示機能

- カード一覧表示.
- カテゴリ一覧表示.

3　内部仕様の策定

アプリケーションの内部を 3 つの構成要素に分けて実装する．

- Model: アプリで使うデータ部分．
- View: ユーザインタフェース部分．
- Controler: View から受け取った入力を処理し，それに応じて Model からデータを抽出して再び View を更新したり，様々なアプリのロジック部分を担当する．
- データとして「カード」と「カテゴリ」が必要． それぞれどの様なデータ構造で表す？
- 「カード」データに必要なフィールド（データ項目）は？
  - カード識別 ID
  - 表，裏: 文字列
  - カテゴリ: カテゴリデータ内の識別コード
  - 作成・更新日付: R の日付オブジェクト
  - 出題回数，正当回数．最終出題日
- 同じように「カテゴリ」データに必要なフィールドをリストアップしてみよう．
- 「カテゴリ」データに必要なフィールドは？
  - カテゴリ名: 文字列
  - カテゴリ・コード: 一意の数字
  - 親カテゴリ・コード
  - カテゴリ・データベースは上記 3 つの列から成る data.frame で表現できる．
- 例えば，あるカテゴリ x に属する下位カテゴリを全て取得したい場合，親カテゴリ・コードが x であるカテゴリを取得すれば良い．

データの構造が明らかになったので，次はこれらデータを操作・管理する関数を定義する．
カード・データベースの操作・管理用関数を定義する．

- 全てのカードを格納するカードデータベースが必要．
  - `create.card.db()`: カードを格納する data.frame を返す．
  - `save.card.db(card.db, filename)`: データベースをファイルに保存する．
  - `load.card.db(filename)`: データベースをファイルからロードした data.frame を返す．

カード・データの操作・管理用関数を定義する．

- カードの作成，削除，更新用関数は更新された `card.db` を返す．
  - `add.card(card.db, front, back, category.id)`
  - `delete.card(card.db, card.id)` 削除するにはカードに一意の ID を付与する必要があることが分かる！
  - `update.card(card.db, card.id, front, back)`
- カード作成時には新規に割り振られたカード ID とデータベースを一緒に返すことにする．

4　カード・データベース管理用関数の実装

以下の 3 つの関数を実装してみよう．実装は全て `flashcard.R` ファイルに書いていく．

- `create.card.db()`: カードを格納する data.frame を返す．
- `save.card.db(card.db, filename)`: データベースをファイルに保存する．
- `load.card.db(filename)`: データベースをファイルからロードする．保存したオブジェクトが作成される．

必要な列を準備した空のデータフレームを作成して返す．

```
create.card.db <- function () {
    data.frame()
}
```

- `save()` はオブジェクト名と一緒にデータを保存し，`load()` はワークスペースの同名の変数を上書きする．
- `saveRDS()` で単一オブジェクトを名前なしで保存し，`readRDS()` で読み込み，指定した変数に代入する．
- ファイルの拡張子は `.rds` を使用するのが慣習．

```
save.card.db <- function (card.db, filename) {
    saveRDS(card.db, file=filename)
}

load.card.db <- function (filename) {
    readRDS(filename)
}
```

データベース管理関数の実装テスト

```
test.db <- create.card.db()
test.db <- rbind(1:5)
test.db

##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    2    3    4    5

save.card.db(test.db, "test.rds")
a.db <- load.card.db("test.rds")
a.db

##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    2    3    4    5
```

5　カード・データ

いよいよカード管理用関数を実装しよう．カード作成関数はカード ID と更新されたデータベースをリストにして返す．削除・更新関数は更新されたカード・データベースを返す．

- `add.card(card.db, front, back, category.id)`
- `update.card(card.db, card.id, front, back)`
- `delete.card(card.db, card.id)`
- データベース内のデータを識別するために一意な ID が必要.
- タイムスタンプ（基準年月日からの秒数）＋乱数を小数点に付加した値とする.
- タイムスタンプを引数に取り，ID を返す関数とする.
- 一様分布から値を 1 つランダム抽出するには `runif(1, min, max)` を使う.

```
generateID <- function(timestamp) {
    floor(timestamp) + runif(1, 0, 1.0)
}
```

5.1 カードの追加

- `add.card()` は，更新したデータベースを `db` に，新規カード ID を `id` に格納したリストを返す.
- 空の `card.db` は 0 行 0 列なので，新しい行はリスト型で列を明示化する.
- 日付は 1970 年 1 月 1 日 0 時 0 分からの秒数で表され，`System.time()` で取得できる.
- 日付は `as.numeric()` 関数で数値に変換して保存する.
- 数値から日付の変換は `as.POSIXlt(time, origin="1970-01-01")`

カードの追加実行例

```
card.db <- create.card.db()
card.db

## data frame with 0 columns and 0 rows

result <- add.card(card.db, "superficial", "うわべだけの，表面的な．", 1)
result$id

## [1] 1508905290

card.db <- add.card(result$db, "paranoid", "被害妄想を持った", 1)$db
card.db <- add.card(card.db, "compiler", "コンパイラ", 1)$db
```

```
card.db

##           ID          表                        裏 カテゴリ      作成日
## 1 1508905290 superficial うわべだけの，表面的な．          1 1508905290
## 2 1508905290    paranoid          被害妄想を持った          1 1508905290
## 3 1508905291    compiler                 コンパイラ          1 1508905290
##   更新日 出題回数 最終出題日
## 1      0        0          0
## 2      0        0          0
## 3      0        0          0

unique(card.db$ID)

## [1] 1508905290 1508905290 1508905291
```

5.2 カードの更新

- card.db から ID が card.id と等しい行の front と back を変更する.
- 「更新日」フィールドのタイムスタンプを更新する.
- 更新したデータベースを返す.

カードの更新実行例

```
card.db <- update.card(card.db, card.db[2, "ID"], "assign",
                       "（変数に）代入する.(仕事を）割り当てる. ")
card.db

##           ID          表                                          裏
## 1 1508905290 superficial                    うわべだけの，表面的な.
## 2 1508905290      assign （変数に）代入する.(仕事を）割り当てる.
## 3 1508905291    compiler                                  コンパイラ
##   カテゴリ      作成日     更新日 出題回数 最終出題日
## 1        1 1508905290          0        0          0
## 2        1 1508905290 1508905290        0          0
## 3        1 1508905290          0        0          0
```

5.3 カードの削除

- カード ID が card.id と一致しないデータを抽出すれば，一致したカードを削除したことになる.
- 削除によって更新されたデータベースを返す.

カードの削除実行例

削除前

```
card.db

##           ID          表                                          裏
## 1 1508905290 superficial                    うわべだけの，表面的な.
## 2 1508905290      assign （変数に）代入する.(仕事を）割り当てる.
## 3 1508905291    compiler                                  コンパイラ
##   カテゴリ      作成日     更新日 出題回数 最終出題日
## 1        1 1508905290          0        0          0
## 2        1 1508905290 1508905290        0          0
## 3        1 1508905290          0        0          0
```

1 枚目のカードを削除する.

```
card.db <- delete.card (card.db, card.db[1, "ID"])
card.db

##           ID       表                                          裏 カテゴリ
## 2 1508905290   assign （変数に）代入する.(仕事を）割り当てる.          1
## 3 1508905291 compiler                                  コンパイラ        1
##       作成日     更新日 出題回数 最終出題日
## 2 1508905290 1508905290        0          0
## 3 1508905290          0        0          0
```

6　次回作業

(1) カテゴリ・データ入力機能.

- カテゴリ・データベースの実装.
- 階層カテゴリの実装.